Last time

- Finish OTR

- Database Security
  - Introduction to Databases
  - Security Requirements
  - Integrity
  - Auditability, Access Control, and Availability
This time

- Database Security
  - Data Inference
  - Statistical Inference
  - Controls against Inference
- Multilevel Security Databases
  - Separation
  - Integrity Locks
  - Designs of MLS Databases
Security vs. Precision

- Security: Forbid any queries that access sensitive data, even if (aggregated) result is no longer sensitive
- Precision: Aggregated result should reveal as much non-sensitive data as possible
Data Inference

- Derivation of sensitive data from non-sensitive data
- Direct attack
  - Attacker issues query that directly yields sensitive data
  - Might obfuscate query to make it less obvious
    - SELECT salary FROM staff WHERE lastname = 'Adams' OR (sex != 'M' AND sex != 'F')
- Indirect attack
  - Infer sensitive data from statistical results
    - As released by governments or pollers
- Data inference is related to data aggregation
  - Derivation of sensitive data from less sensitive data
Statistical Inference Attacks

• Sum
  •Leaks sensitive data if sum covers only one record or if attacker can control set of covered records
    • SELECT SUM(salary)
    • SELECT SUM(salary) WHERE lastname != 'Adams'

• Count
  • Useful in attack above

• Mean
  • sum = count * mean

• Median
  • Intersecting medians might leak sensitive data
  • See text for example
Tracker Attacks

• Focus on queries of type \texttt{SUM} or \texttt{COUNT}

• Assume that DBMS refuses to answer a query if number of matching records is smaller than \(k\) or larger than \(N-k\) (Why?)
  – \(N\): number of records in database

• A tracker \(T\) is a query whose result matches between \(2k\) and \(N-2k\) records
  – DBMS will answer \(T\) (and \textit{not} \(T\))

• Assume that there is a query \(C\) that DBMS refuses to answer since it matches fewer than \(k\) or more than \(N-k\) records
Tracker Attacks (cont.)

- Let \( q() \) be the result of a query and \( S \) the set of all records.

- Using Venn diagrams, we can show that
  - \( q(C) = q(C \text{ or } T) + q(C \text{ or } \text{not } T) - q(S) \)
  - Use right-hand side for computing \( q(C) \) if \( q(C) \) matches fewer than \( k \) records.
  - \( q(C) = 2 \times q(S) - q(\text{not } C \text{ or } T) - q(\text{not } C \text{ or } \text{not } T) \)
  - Use right-hand side for computing \( q(C) \) if \( q(C) \) matches more than \( n-k \) records.

- In general, simple logic or linear algebra might allow an attacker to convert a forbidden query into multiple, allowed queries.
Controls for Statistical Inference Attacks

• Apply control to query or to data items
  – As seen, former is difficult

• Suppression and concealing are two controls applied to data items

• Suppression
  – Suppress sensitive data from result

• Concealing
  – Answer is close to actual value, but not exactly
Controls (cont.)

- n-item k-percent rule
  - If there are n records and they represent over k percent of reported result, omit these records from result
  - However, omission itself might leak information or omitted value could be derived with other means

- Combined results
  - Report set or range of possible values

- Random sample
  - Compute result on random sample of database
  - Need to use same sample for equivalent queries
Controls (cont.)

● Random data perturbation
  - Add or subtract small random error to/from each value before computing result
  - Expectation is that statistical properties are maintained

● Query analysis
  - Maintain history of user’s queries and observe possible inferences
  - Costly, fails for colluding users
Aggregation

- Building sensitive results from less sensitive inputs, typically from different sources (e.g., people)
- Aggregation can take place outside of a DBMS, which makes it difficult to control
  - People talking to each other
- Closely related to data mining (see later), where information from different databases is combined
Multilevel Security (MLS) Databases

• Support classification/compartmentalization of information according to its confidentiality/integrity
  – Two levels (sensitive and not sensitive) might not be sufficient

• At element level if necessary
  – Salary might be sensitive only for some employees
  – Other information in employee’s record might not be sensitive

• In an MLS database, each object has a classification and maybe a compartment
  – Object can be element, aggregate, column, or row
*-Property

- Implementing the *-property in an MLS database is difficult
  - User doing a write-up though user cannot read data at higher level (Blind writes)
  - Write-downs need a sanitization mechanism
  - Trusted processes that can do anything
- DBMS must have read and write access at all levels to answer user queries, perform back-ups, optimize database,…
  - Must trust DBMS
Confidentiality

- Depending on a user’s level, he/she might get different answers for a query
  - Less precision for low-level users
- Existence of a record itself could be confidential
- Keeping existence hidden can lead to having multiple records with the same primary key (polyinstantiation)
  - Admin notices that there is no record for Bob Hill and creates one
  - However, Bob Hill is a secret agent, so there already is a record, which admin cannot see
  - DBMS must allow admin’s request, else admin would get suspicious
Partitioning

- Have separate database for each classification level
- Simple, often used in practice
- Might lead to data stored redundantly in multiple databases
- Doesn’t address the problem of a high-level user needing access to low-level data combined with high-level data
Encryption

- Separate data by encrypting it with a key unique to its classification level
- Must be careful to use encryption scheme in the right way
  - E.g., encrypting the same value in different record with the same key should lead to different ciphertexts
- Processing of a query becomes expensive, many records might have to be decrypted
  - Doing the processing directly on the encrypted data is an active research area
Integrity Lock

- Provides both integrity and access control
- Each data item consists of
  - The actual data item
  - A sensitivity label (maybe concealed)
  - A cryptographic signature (or MAC) covering the above plus the item’s attribute name and its record number
- Signature protects against attacks on the above fields, such as attacks trying to modify the sensitivity label, and attacks trying to move/copy the item in the database
- This scheme does not protect against replay attacks
Integrity Lock (cont.)

- Any (untrusted) database can be used to store data items and their integrity locks
  - Locks can consume lots of space (maybe multiple locks per record)
- (Trusted) procedure handles access control and manages integrity locks
  - E.g., updates sensitivity level to enforce *-property or re-computes signature after a write access
  - Expensive
- Have to encrypt items and locks if there are other ways to get access to data in database
  - Makes query processing even more expensive
Trusted Front End

- Front end authenticates a user and forwards user query to old-style DBMS
- Front end gets result from DBMS and removes data items that user is not allowed to see
- Allows use of existing DBMS and databases
- Inefficient if DBMS returns lots of items and most of them are being dropped by front end
Commutative Filters

- Front end re-writes user query according to a user’s classification
  - Remove attributes that user is not allowed to see
  - Add constraint expressing user’s classification
- Benefits from DBMS’ superior query processing capabilities and discards forbidden data items early on
- Front end might still have to do some post processing
Distributed/Federated Databases

- Based on partitioning
- Front end forwards user query only to databases that user can access based on classification
- Front end might have to combine the results from multiple databases
  - Complex process, front end essentially becomes a DBMS
- Doesn’t scale to lots of classification levels
Views

- Many DBMS support views
- A view is a logical database that represents a subset of some other database
  - `CREATE VIEW foo AS SELECT * FROM bar WHERE...`
- Element in view can correspond to an element in underlying database or be a combination of multiple elements
  - E.g., their sum
- Views can be used for access control
  - A user’s view of a database consists of only the data that the user is allowed to access
  - Hide attribute/row unless user is allowed to access at least one element, set to UNDEFINED any elements that user can’t access
Truman vs. Non-Truman semantics

- **Truman semantics:** the DBMS pretends that the data the user is allowed to access is all the data there is
  - Like “The Truman Show”
  - All queries will succeed, even if they return imprecise results

- **Non-Truman semantics:** the DBMS can reject queries that ask for data the user is not allowed to access
  - Any queries that succeed will produce precise answers
  - Some queries will fail
Recap

• Database Security
  – Data Inference
  – Statistical Inference
  – Controls against Inference

• Multilevel Security Databases
  – Separation
  – Integrity Locks
  – Designs of MLS Databases
Next time

- Data Mining
  - Integrity and Availability
  - Privacy and Data Mining
  - Privacy-Preserving Data Mining